Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 9994015, 9 pages
https://doi.org/10.1155/2021/9994015

Research Article

An Accelerated Proximal Algorithm for the Difference of Convex Programming

Feichao Shen,[1] Ying Zhang,[2] and Xueyong Wang[3]

[1]*School of Mathematics & Computing Science, Guilin University of Electronic Technology, Guilin, Guangxi 541004, China*
[2]*School Basic Teaching, Shandong Water Conservancy Vocat Coll, Rizhao, Shandong 276800, China*
[3]*School of Management, Qufu Normal University, Rizhao, Shandong 276800, China*

Correspondence should be addressed to Xueyong Wang; yonggk@163.com

Received 4 March 2021; Revised 8 April 2021; Accepted 15 April 2021; Published 26 April 2021

Academic Editor: Zhenbo Wang

In this paper, we propose an accelerated proximal point algorithm for the difference of convex (DC) optimization problem by combining the extrapolation technique with the proximal difference of convex algorithm. By making full use of the special structure of DC decomposition and the information of stepsize, we prove that the proposed algorithm converges at rate of $O(1/k^2)$ under milder conditions. The given numerical experiments show the superiority of the proposed algorithm to some existing algorithms.

1. Introduction

Difference of convex problem (DCP) is an important kind of nonlinear programming problems in which the objective function is described as the difference of convex (DC) functions. It finds numerous applications in digital communication system [1], assignment and power allocation [2], compressed sensing [3–6], and so on [7–13].

It is well known that the method to solve the DCP is the so-called difference of the convex algorithm (DCA) [14] in which the concave part is replaced by a linear majorant in the objective function and a convex optimization subproblem needs to be solved at each iteration. Note that the difficulty of the involved subproblem relies heavily on the DC decomposition of the objective function, and it can be easily solved when the objective function can be written as the sum of a smooth convex function with Lipschitz gradient, a proper closed convex function, and a continuous concave function [15]. Motivated by this, Gotoh et al. [16] proposed the so-called proximal difference of the convex algorithm (PDCA) for solving DCP, in which not only the concave part is replaced by a linear majorant in each iteration but also the smooth convex part is replaced by some techniques. Furthermore, if the proximal mapping of the proper closed convex function can be easily computed, then the subproblem involved in the PDCA can be solved efficiently. However, when the concave part of the objective is void, the PDCA reduces to the proximal gradient algorithm which may be slow in computing [17]. In fact, since the convergence rate of the PDCA heavily depends on the Lojasiewicz exponent of the objective function, the PDCA converges linearly in general [18, 19]. To accelerate the convergence rate of the proximal difference of the convex algorithm, researchers recall the well-known extrapolation technique to design some efficient algorithms [20–24]. This technique has been extensively used in accelerating the proximal type algorithms for convex programming [25, 26], and the convergence rate of the algorithms can be improved from $O(1/k)$ to $O(1/k^2)$. Motivated by this, Wen et al. [27] proposed the proximal difference of the convex algorithm with extrapolation (PDCAE) for solving the DCP. The numerical experiments [27] show that the PDCAE has a better performance although it converges linearly in theory [27]. Now, a question is posed naturally: can we propose new type of the PDCA in which the convergence rate can be improved in theory? This constitutes the motivation of the paper.

In this paper, inspired by the work in [20–23, 27], we establish an accelerated proximal DC programming

algorithm (APDCA) for the DCP by combining the extrapolation technique and the PDCA. In the algorithm, the current iteration point is replaced by a linear combination of the previous two points, and extrapolation technique is involved in the stepsize. By making full use of the special structure of DC decomposition and the information of stepsize, we prove that the APDCA converges at rate of $O(1/k^2)$ under milder conditions. The given numerical experiments show the superiority to some existing algorithms.

The remainder of the paper is organized as follows. In Section 2, we describe the DC optimization problem considered in this paper and present our new designed algorithm. In Section 3, we establish the global convergence and the quadratic convergence rate of the new designed algorithm. Some numerical experiments are provided in Section 4. Some conclusions are drawn in Section 5.

To end this section, we recall some definitions used in the subsequent analysis [28–30].

For an extended real valued function $f: R^n \longrightarrow [-\infty, +\infty]$, we denote its domain by dom $f = \{x \in R^n: f(x) < +\infty\}$. The function f is said to be strongly convex if there exists an $a > 0$ such that $\nabla^2 f(x) \geq aI$ for all $x \in S$, where S is a convex set and I is a identity matrix. The function f is said to be proper if it never equals $-\infty$ and dom $f \neq \varnothing$. Moreover, a proper function is closed if it is lower semicontinuous. A proper closed function f is said to be level-bounded if the lower level sets of f are bounded; that is, $\{x \in R^n: f(x) \geq r\}$ are bounded for any $r \in R$. Given a proper closed function $f: R^n \longrightarrow R \cup \{+\infty\}$, the limiting subdifferential of f at $x \in \mathrm{dom} f$ is given as follows:

$$\partial f(x) = \left\{ v \in R^n: \exists x_k \xrightarrow{f} x,\ v_k \longrightarrow v \text{ with } \liminf_{y \longrightarrow x_k} \frac{f(y) - f(x_k) - \langle v_k, y - x_k \rangle}{\|y - x_k\|} \geq 0, \quad k \right\}, \tag{1}$$

where $z \longrightarrow^f x$ mean $z \longrightarrow x$ and $f(z) \longrightarrow f(x)$. Note that dom $\partial f = \{x \in R^n: \partial f(x) \neq \varnothing\}$. It is well known that the (limiting) subdifferential reduces to the classical subdifferential in convex analysis when f is a convex function; that is,

$$\partial f(x) = \{v \in R^n: f(u) - f(x) - \langle v, u - x \rangle \geq 0, \quad u \in R^n\}. \tag{2}$$

Furthermore, if f is continuously differentiable, then the (limiting) subdifferential reduces to the gradient of f and denoted by ∇f.

2. Algorithms for DC Programming

Consider the following difference of convex programming:

$$\min_{x \in R^n} \{F(x) := f(x) + g(x) - h(x)\}, \tag{3}$$

where $f: R^n \longrightarrow R$ is a strongly convex function with $a > 0$, $g: R^n \longrightarrow R$ is a smooth convex function, ∇g is Lipschitz continuous with $L_g > 0$, $h: R^n \longrightarrow R$ is a continuous convex function, and ∇h is Lipschitz continuous with $L_h > 0$.

For the DCP, the following is a classical DCA which takes the following iterative scheme [14]:

$$x_{k+1} \in \arg\min_{x \in R^n} f(x) + g(x) - \langle \nabla h(x_k), x - x_k \rangle. \tag{4}$$

By replacing the concave part in the objective function by a linear majorant and replacing the smooth convex part by a quadratic majorant, Gotoh et al. [16] proposed a proximal DCA for the DCP. For the sake of completeness, we list Algorithm 1 as follows.

Despite a simple subproblem is involved in the algorithm, the PDCA is potentially slow [19, 27]. To accelerate the convergence rate of the PDCA, we incorporate extrapolation technique into the PDCA to obtain the following algorithm (Algorithm 2).

3. Convergence Analysis of the APDCA

In this section, we establish the global convergence of the algorithm and its convergence rate. To continue, we first recall the following conclusions.

Lemma 1 (see [25]). *Let f be a continuously differentiable function with Lipschitz continuity gradient whose Lipschitz constant $L(f) > 0$. Then, for any $L \geq L(f)$, it holds that*

$$f(x) \leq f(y) + \langle \nabla f(y), x - y \rangle + \frac{L}{2}\|x - y\|^2, \quad x, y \in R^n. \tag{5}$$

Lemma 2. *Let $\mu \geq (1/2a)$. For the sequence $\{x_k, y_k\}$ generated by the APDCA, it holds that*

$$\mu(F(x) - F(x_{k+1})) \geq \|x - x_{k+1}\|^2 - \|x - y_k\|^2, \quad k \geq 1. \tag{6}$$

Proof. Since f is strong convex function, there exists constant $a > 0$ such that

$$f(x) \geq f(x_{k+1}) + \langle \xi_{k+1}, x - x_{k+1} \rangle + a\|x - x_{k+1}\|^2, \tag{7}$$

where $\xi_{k+1} \in \partial f(x_{k+1})$.

Connecting the fact that $\nabla h(x)$ is Lipschitz continuous with constant $L_h > 0$ with Lemma 1, we have

$$h(x) \leq h(y_k) + \langle \nabla h(y_k), x - y_k \rangle + \frac{1}{2\mu}\|x - y_k\|^2, \tag{8}$$

where $0 < \mu \leq (1/L_h)$, which means that

$$-h(x) \geq -h(y_k) - \langle \nabla h(y_k), x - y_k \rangle - \frac{1}{2\mu}\|x - y_k\|^2. \tag{9}$$

Initial step. Take $\varepsilon > 0$, $\mu = (1/L_g)$, and $x_0 \in \mathrm{dom} f$.
Iterative step. Compute the new iterate by the following iterative scheme:
$x_{k+1} \in \arg\min_{x \in R^n} f(x) + g(x_k) - h(x_k) - \langle \nabla g(x_k) - \nabla h(x_k), x - x_k \rangle + 1/2\mu \|x - x_k\|^2$
until $\|x_{k+1} - x_k\| \le \varepsilon$ is satisfied
where $L_g > 0$ is the Lipschitz constant of ∇g.

Algorithm 1: PDCA.

Initial step. Take $0 < \mu \le (1/\max\{L_g, L_h\})$, $\{\beta_k\} \subset [0, 1)$ with $0 \le \sup_k \beta_k < 1$, $\varepsilon > 0$, $t_0 = t_1 = 1$, and $x_1 = x_0 \in \mathrm{dom} f$.
Iterative step. Compute the new iterate by the following iterative scheme:
$\beta_k = t_k - 1/t_{k+1}$ and $t_{k+1} = 1 + \sqrt{1 + 4t_k^2}/2$
$y_k = x_k + \beta_k (x_k - x_{k-1})$,
$x_{k+1} = \arg\min_{x \in R^n} \{ f(x) + g(y_k) - h(y_k) + \langle \nabla g(y_k) - \nabla h(y_k), x - y_k \rangle + 1/2\mu \|x - y_k\|^2 \}$
until $\|x_{k+1} - x_k\| \le \varepsilon$ is satisfied.

Algorithm 2: APDCA.

It follows from g is convex function that

$$g(x) \ge g(y_k) + \langle \nabla g(y_k), x - y_k \rangle. \tag{10}$$

Connecting (7) and (9) with (10), we have

$$f(x) + g(x) - h(x) \ge f(x_{k+1}) + g(y_k) - h(y_k) + \langle \xi_{k+1}, x - x_{k+1} \rangle + \langle \nabla g(y_k) - \nabla h(y_k), x - y_k \rangle + a\|x - x_{k+1}\|^2 - \frac{1}{2\mu}\|x - y_k\|^2. \tag{11}$$

On the other hand, since h is convex, it follows that

$$h(x) \ge h(y_k) + \langle \nabla h(y_k), x - y_k \rangle, \tag{12}$$

which means that

$$-h(x) \le -h(y_k) - \langle \nabla h(y_k), x - y_k \rangle. \tag{13}$$

Connecting the fact that $\nabla g(x)$ is Lipschitz continuous with constant $L_h > 0$ with Lemma 1, we have

$$g(x) \le g(y_k) + \langle \nabla g(y_k), x - y_k \rangle + \frac{1}{2\mu}\|x - y_k\|^2, \tag{14}$$

where $0 < \mu \le (1/L_g)$. Summing (13) and (14), we have

$$g(x) - h(x) \le g(y_k) - h(y_k) + \langle \nabla g(y_k) - \nabla h(y_k), x - y_k \rangle + \frac{1}{2\mu}\|x - y_k\|^2. \tag{15}$$

Adding $f(x)$ to both sides of (15) yields

$$f(x) + g(x) - h(x) \le f(x) + g(y_k) - h(y_k) + \langle \nabla g(y_k) - \nabla h(y_k), x - y_k \rangle + \frac{1}{2\mu}\|x - y_k\|^2. \tag{16}$$

By taking $x = x_{k+1}$, (16) yields that

$$f(x_{k+1}) + g(x_{k+1}) - h(x_{k+1}) \le f(x_{k+1}) + g(y_k) - h(y_k) + \langle \nabla g(y_k) - \nabla h(y_k), x_{k+1} - y_k \rangle + \frac{1}{2\mu}\|x_{k+1} - y_k\|^2. \tag{17}$$

By the optimality conditions of (8), one has

$$\xi_{k+1} + \nabla g(y_k) - \nabla h(y_k) + \frac{1}{\mu}(x_{k+1} - y_k) = 0, \tag{18}$$

that is,

$$-\frac{1}{\mu}(x_{k+1} - y_k) = \xi_{k+1} + \nabla g(y_k) - \nabla h(y_k). \tag{19}$$

Then, for $0 < \mu \le (1/\max\{L_g, L_h\})$, it follows from (11) and (17) that

$$\begin{aligned} F(x) - F(x_{k+1}) &\ge \langle \xi_{k+1} + \nabla g(y_k) - \nabla h(y_k), x - x_{k+1}\rangle + a\|x - x_{k+1}\|^2 - \frac{1}{2\mu}\|x - y_k\|^2 - \frac{1}{2\mu}\|y_k - x_{k+1}\|^2 \\ &= -\frac{1}{\mu}\langle x_{k+1} - y_k, x - x_{k+1}\rangle + a\|x - x_{k+1}\|^2 - \frac{1}{2\mu}\|x - y_k\|^2 - \frac{1}{2\mu}\|y_k - x_{k+1}\|^2 \\ &= \frac{1}{2\mu}\left(\|y_k - x_{k+1}\|^2 + \|x - x_{k+1}\|^2 - \|x - y_k\|^2\right) + a\|x - x_{k+1}\|^2 - \frac{1}{2\mu}\|x - y_k\|^2 - \frac{1}{2\mu}\|y_k - x_{k+1}\|^2 \\ &= \frac{1}{2\mu}\left((1 + 2a\mu)\|x - x_{k+1}\|^2 - 2\|x - y_k\|^2\right) \\ &\ge \frac{1}{\mu}\left(\|x - x_{k+1}\|^2 - \|x - y_k\|^2\right), \end{aligned} \tag{20}$$

where the first equality follows from (19), the second equality follows from the fact that $2\langle a - b, a - c\rangle = \|a - c\|^2 + \|a - b\|^2 - \|b - c\|^2$, $a, b, c \in R^n$, and the last inequality follows from $2a\mu \ge 1$. We have conclusion (6).

Before proceeding further, we need the following conclusions. □

Lemma 3 (see [25, 31]). *Let $t_0 = t_1 = 1$. Then, the sequence $\{t_k\}$ generated by (6) is increasing, and $t_k \ge ((k + 1)/2)$.*

Lemma 4. *Let $\{x_k, y_k\}$ be a sequence generated by the APDCA. Then,*

$$\mu\left(t_k^2 v_k - t_{k+1}^2 v_{k+1}\right) \ge \|u_{k+1}\|^2 - \|u_k\|^2, \tag{21}$$

where $u_k = t_k x_k - (t_k - 1)x_{k-1} - x^*$, $v_k = F(x_k) - F(x^*)$, and x^* is the critical point of problem (3).

Proof. From (7) and (6), we have $y_k = x_k + ((t_k - 1)/(t_{k+1}))(x_k - x_{k-1})$. Then, it follows that

$$\begin{aligned} \|u_{k+1}\|^2 - \|u_k\|^2 &= \|t_{k+1}x_{k+1} - (t_{k+1} - 1)x_k - x^*\|^2 - \|t_k x_k - (t_k - 1)x_{k-1} - x^*\|^2 \\ &= \|t_{k+1}x_{k+1} - (t_{k+1} - 1)x_k - x^*\|^2 - \|t_{k+1}y_k - (t_{k+1} - 1)x_k - x^*\|^2. \end{aligned} \tag{22}$$

Hence, to show the assertion, we only need to show that

$$\mu\left(t_k^2 v_k - t_{k+1}^2 v_{k+1}\right) \ge \|t_{k+1}x_{k+1} - (t_{k+1} - 1)x_k - x^*\|^2 - \|t_{k+1}y_k - (t_{k+1} - 1)x_k - x^*\|^2. \tag{23}$$

In fact, by taking $x = x_k$, one has from Lemma 2 that

$$\mu(F(x_k) - F(x_{k+1})) \ge \|x_{k+1} - x_k\|^2 - \|x_k - y_k\|^2. \tag{24}$$

Hence,

$$\mu(v_k - v_{k+1}) \ge \|x_{k+1} - x_k\|^2 - \|x_k - y_k\|^2. \tag{25}$$

Using Lemma 2 again, one has from $x = x^*$ that

$$\mu(F(x^*) - F(x_{k+1})) \ge \|x_{k+1} - x^*\|^2 - \|y_k - x^*\|^2, \tag{26}$$

that is,

$$-\mu v_{k+1} \ge \|x_{k+1} - x^*\|^2 - \|y_k - x^*\|^2. \tag{27}$$

Multiplying (25) by t_k^2 and (27) by t_{k+1}, respectively, and summing them yield

$$\begin{aligned}\mu\left(t_k^2 v_k - t_{k+1}^2 v_{k+1}\right) &= \mu\left(t_k^2 v_k - \left(t_k^2 + t_{k+1}\right)v_{k+1}\right)\\ &\geq t_k^2\left(\|x_{k+1} - x_k\|^2 - \|x_k - y_k\|^2\right) + t_{k+1}\left(\|x_{k+1} - x^*\|^2 - \|y_k - x^*\|^2\right)\\ &= \left(t_{k+1}^2 - t_{k+1}\right)\left(\|x_{k+1} - x_k\|^2 - \|x_k - y_k\|^2\right) + t_{k+1}\left(\|x_{k+1} - x^*\|^2 - \|y_k - x^*\|^2\right)\\ &= \|t_{k+1}x_{k+1} - (t_{k+1} - 1)x_k - x^*\|^2 - \|t_{k+1}y_k - (t_{k+1} - 1)x_k - x^*\|^2,\end{aligned} \tag{28}$$

where the first equality follows from the fact that $t_{k+1}^2 = t_k^2 + t_{k+1}$ and the last equality follows by some manipulation. The desired result follows.

Now, we are ready to show the convergence rate of the APDCA. □

Theorem 1. *For the sequence $\{x_k\}$ generated by the APDCA, it holds that*

$$F(x_k) - F(x^*) \leq \frac{4\|x_0 - x^*\|^2}{\mu(k+1)^2}, \tag{29}$$

where x^* is a stationary point of (3).

Proof. Using the notations used in Lemma 4, let $k = 0$, and it follows from (27) that

$$-\mu v_1 \geq \|x_1 - x^*\|_2 - \|y_0 - x^*\|^2. \tag{30}$$

Hence,

$$\mu v_1 + \|x_1 - x^*\|^2 \leq \|y_0 - x^*\|^2. \tag{31}$$

Then, from Lemma 4, we know that the sequence $\{\mu t_k^2 v_k + \|u_k\|^2\}$ is nonincreasing. Therefore,

$$\mu t_k^2 v_k \leq \mu t_k^2 v_k + \|u_k\|^2 \leq \mu t_1^2 v_1 + \|x_1 - x^*\|^2 \leq \|y_0 - x^*\|^2 = \|x_0 - x^*\|^2, \tag{32}$$

where the second inequation follows from $t_1 = 1$ and $u_1 = x_1 - x^*$, and the last equation follows from $t_1 = 1$.

Then, it follows from Lemma 3 that

$$F(x_k) - F(x^*) \leq \frac{4\|x_0 - x^*\|^2}{\mu(k+1)^2}. \tag{33}$$

The desired result follows. □

4. Numerical Experiments

In this section, we evaluate the performance of the APDCA by applying it to the DC regularized least squares problem. We will compare the performance of the APDCA with the algorithm in [15] (PDCA) and GIST in [32].

On APDCA and PDCA, we set $(1/\mu) = L_g = \lambda_{\max}(A^T A)$ and $c = (L_g/2)$. On GIST, we set $\sigma = 10^{-5}$, $m = 5$, $\eta = 2$, and $(1/\alpha_{\min}) = \alpha_{\max} = 10^{30}$. We initialize the three algorithms at the origin point and terminate the algorithms when

$$\frac{\|x_k - x_{k-1}\|}{\max\{1, \|x_k\|\}} < 10^{-5}. \tag{34}$$

Furthermore, we terminate PDCA when the number of iteration is more than 5000 (denoted by "max" on the report).

Example 1. Least squares problems with l_{1-2} regularizer are as follows:

$$\min_{x \in R^n}\left\{\frac{1}{2}\|Ax - b\|^2 + c\|x\|^2 + \lambda\|x\|_1 - c\|x\|^2 - \lambda\|x\|\right\}, \tag{35}$$

where $A \in R^{m \times n}$, $b \in R^m$, $c > 0$, and $\lambda > 0$ is the regularization parameter.

This problem takes the form of (3) with $f(x) = c\|x\|^2 + \lambda\|x\|_1$, $g(x) = (1/2)\|Ax - b\|^2$, and $h(x) = c\|x\|^2 + \lambda\|x\|$. Note that the purpose of adding $c\|x\|^2$ is to ensure strong convexity of $f(x)$.

To compare the performance of the three algorithms, we report the number of iterations (denoted by Iter), CPU times in seconds (denoted by CPU time), the sparsity of the solution (denoted by sparsity), and the function values at termination (denoted by fval), averaged over the 30 random instances. The numerical results are reported in Tables 1 and 2, from which we can see that the APDCA always outperforms PDCA and GIST. Specifically, from Table 1, we can see that the APDCA is about 2.5 times faster than GIST and is about 5.2 times faster than PDCA for the parameter $\lambda = 5 \times e^{-4}$. From Table 2, we can see that the APDCA is about 2.1 times faster than GIST and is about 8.4 times faster than PDCA for the parameter $\lambda = 1 \times e^{-3}$. Tables 1 and 2 also show that the APDCA requires fewer iteration steps than the other two

Table 1: Solving (35) on random instances, $\lambda = 5 \times e^{-4}$.

		Iter			CPU time		
m	n	GIST	APDCA	PDCA	GIST	APDCA	PDCA
720	2560	1750	909	Max	3.57	1.38	7.37
1440	5120	1629	802	Max	13.7	5.0	31.8
2160	7680	1724	802	Max	28.5	10.0	62.2
2880	10240	1742	1002	Max	52.8	22.3	112.2
3600	12800	1799	1002	Max	83.8	34.3	174.7
4320	15360	1739	1002	Max	113.7	48.9	246.5
5040	17920	1778	1002	Max	160.7	66.9	334.5
5760	20480	1826	1002	Max	178.3	71.5	366.1
6480	23040	1778	975	Max	244.3	100.5	524.1
7200	25600	1752	975	Max	317.4	130.9	692.6
		Sparsity			Fval		
m	n	GIST	APDCA	PDCA	GIST	APDCA	PDCA
720	2560	783	761	1132	2.9755e−2	2.9743e−2	4.5442e−2
1440	5120	1575	1614	2240	6.1144e−2	6.1122e−2	9.4466e−2
2160	7680	2367	2424	3425	9.4648e−2	9.4612e−2	1.4594e−1
2880	10240	3117	2910	4496	1.2312e−1	1.2308e−1	1.8319e−1
3600	12800	3889	3644	5707	1.5896e−1	1.5890e−1	2.4309e−1
4320	15360	4766	4376	6720	1.8879e−1	1.8869e−1	2.8401e−1
5040	17920	5497	5141	7911	2.2523e−1	2.2512e−1	3.4175e−1
5760	20480	6327	5931	9181	2.6870e−1	2.6859e−1	4.1224e−1
6480	23040	7065	6716	10184	2.9070e−1	2.9098e−1	4.3889e−1
7200	25600	7865	7449	11286	3.2206e−1	3.2191e−1	4.8588e−1

Table 2: Solving (35) on random instances, $\lambda = 1 \times e^{-3}$.

		Iter			CPU time		
m	n	GIST	APDCA	PDCA	GIST	APDCA	PDCA
720	2560	972	591	Max	1.5	0.7	5.4
1440	5120	968	602	Max	6.1	2.8	23.2
2160	7680	993	602	Max	13.6	6.1	50.2
2880	10240	835	602	Max	19.8	10.6	88.6
3600	12800	973	602	Max	36.1	16.7	139.8
4320	15360	931	602	Max	49.2	23.5	202.5
5040	17920	941	602	Max	67.5	32.6	296.4
5760	20480	979	602	Max	100.7	43.5	354.9
6480	23040	992	602	Max	116.3	54.9	449.8
7200	25600	939	602	Max	138.0	67.4	558.5
		Sparsity			Fval		
m	n	GIST	APDCA	PDCA	GIST	APDCA	PDCA
720	2560	728	703	927	6.2438e−2	6.2430e−2	7.6433e−2
1440	5120	1449	1381	1838	1.3160e−1	1.3159e−1	1.6346e−1
2160	7680	2168	2086	2810	2.0060e−1	2.0058ee−1	2.5146e−1
2880	10240	2853	2745	3618	2.3976e−1	2.3973e−1	2.7654e−1
3600	12800	3675	3557	4607	3.0264e−1	3.0260e−1	3.5620e−1
4320	15360	4368	4195	5523	3.9802e−1	3.9798e−1	4.7740e−1
5040	17920	5132	4925	6501	4.7413e−1	4.7407e−1	5.7676e−1
5760	20480	5825	5656	7358	5.3208e−1	5.3202e−1	6.3891e−1
6480	23040	6597	6311	8361	5.7707e−1	5.7699e−1	6.9385e−1
7200	25600	7270	7052	9269	6.4648e−1	6.4640e−1	7.7325e−1

algorithms. Specifically, from Table 1, the iteration step of APDCA is about 53% of GIST for the parameter $\lambda = 5 \times e^{-4}$. From Table 2, the iteration step of APDCA is about 64% of GIST for the parameter $\lambda = 1 \times e^{-3}$. Meanwhile, Tables 1 and 2 also show that the solution given by APDCA is more sparse than that given by GIST and PDCA.

Example 2. Least squares problems with logarithmic regularizer are as follows:

Table 3: Solving (35) on random instances, $\lambda = 5 \times e^{-4}$.

			Iter			CPU time	
m	n	GIST	APDCA	PDCA	GIST	APDCA	PDCA
720	2560	843	596	Max	1.6	0.7	5.5
1440	5120	672	602	Max	5.6	3.0	22.3
2160	7680	873	602	Max	12.4	6.1	49.5
2880	10240	876	602	Max	21.2	10.6	87.4
3600	12800	871	602	Max	32.7	16.6	138.0
4320	15360	845	602	Max	45.1	23.4	194.8
5040	17920	872	602	Max	62.5	32.0	265.6
5760	20480	846	602	Max	79.4	41.4	345.3
6480	23040	877	602	Max	104.1	52.8	441.0
7200	25600	816	602	Max	120.4	66.0	547.5
			Sparsity			Fval	
m	n	GIST	APDCA	PDCA	GIST	APDCA	PDCA
720	2560	705	661	931	3.8979e−2	3.8973e−2	5.6815e−2
1440	5120	1395	1345	1794	7.1306e−2	7.1293e−2	9.3006e−2
2160	7680	2123	2011	2710	1.1455e−1	1.1453e−1	1.5861e−1
2880	10240	2809	2705	3597	1.4878e−1	1.4876e−1	2.0601e−1
3600	12800	3570	3418	4503	1.9187e−1	1.9182e−1	2.7236e−1
4320	15360	4277	4103	5370	2.3163e−1	2.3159e−1	3.1699e−1
5040	17920	5042	4729	6287	2.6491e−1	2.6486e−1	3.6295e−1
5760	20480	5689	5501	7199	3.0649e−1	3.0643e−1	4.3049e−1
6480	23040	6353	6093	8057	3.4115e−1	3.4110e−1	4.7749e−1
7200	25600	7139	6089	8924	3.7435e−1	3.7427e−1	5.1004e−1

Table 4: Solving (40) on random instances, $\lambda = 1 \times e^{-3}$.

			Iter			CPU time	
m	n	GIST	APDCA	PDCA	GIST	APDCA	PDCA
720	2560	497	329	4658	0.9	0.4	5.2
1440	5120	468	402	4582	3.1	1.9	20.5
2160	7680	496	402	4739	6.8	4.0	46.8
2880	10240	472	402	4527	11.1	7.0	79.5
3600	12800	494	402	4601	18.3	11.1	126.5
4320	15360	505	402	4602	26.6	16.5	179.0
5040	17920	451	402	4428	31.8	21.3	234.7
5760	20480	448	402	4446	41.2	27.7	304.2
6480	23040	459	402	4602	52.8	35.0	403.6
7200	25600	487	402	4668	70.7	44.0	510.4
			Sparsity			Fval	
m	n	GIST	APDCA	PDCA	GIST	APDCA	PDCA
720	2560	628	635	658	7.5032e−2	7.5032e−2	7.5053e−2
1440	5120	1300	1248	1337	1.4892e−1	1.4891e−1	1.4896e−1
2160	7680	1987	1865	1965	2.3348e−1	2.3347e−1	2.3354e−1
2880	10240	2543	2462	2627	3.0410e−1	3.0410e−1	3.0416e−1
3600	12800	3156	3072	3252	3.8829e−1	3.8828e−1	3.8837e−1
4320	15360	3831	3703	3973	4.5346e−1	4.5344e−1	4.5348e−1
5040	17920	4460	4300	4605	5.2664e−1	5.2662e−1	5.2676e−1
5760	20480	5124	4991	5268	5.9404e−1	5.9402e−1	5.9417e−1
6480	23040	5761	5540	5919	6.8740e−1	6.8737e−1	6.8756e−1
7200	25600	6365	6231	6632	7.6681e−1	7.6678e−1	7.6700e−1

$$\min_{x \in R^n} \left\{ \frac{1}{2}\|Ax - b\|^2 + c\|x\|^2 + \sum_{i=1}^{n} \left[\lambda \log\left(|x_i| + \varepsilon\right) - \lambda \log \varepsilon \right] - c\|x\|^2 \right\}, \tag{36}$$

where $A \in R^{m \times n}, b \in R^m, \varepsilon > 0$ is a constant, and $\lambda > 0$ is the regularization parameter.

This problem takes the form of (3) with $f(x) = c\|x\|^2 + (\lambda/\varepsilon)\|x\|_1$, $g(x) = (1/2)\|Ax - b\|^2$, and $h(x) = c\|x\|^2 + \sum_{i=1}^{n}$

$\lambda[(|x_i|/\varepsilon) - \log(|x_i| + \varepsilon) + \log\varepsilon]$. Note that the purpose of adding $c\|x\|^2$ is to ensure strong convexity of $f(x)$. For this example, we set $\epsilon = 0.5$.

To compare the performance of the three algorithms, we report the number of iterations (denoted by Iter), CPU times in seconds (denoted by CPU time), the sparsity of the solution (denoted by sparsity), and the function values at termination (denoted by fval), averaged over the 30 random instances. The numerical results are reported in Tables 3 and 4, from which we can see that the APDCA always outperforms PDCA and GIST. Specifically, from Table 3, we can see that the APDCA is about 1.9 times faster than GIST and is about 8.3 times faster than PDCA for the parameter $\lambda = 5 \times e^{-4}$. From Table 4, we can see that the APDCA is about 1.6 times faster than GIST and is about 11.3 times faster than PDCA for the parameter $\lambda = 1 \times e^{-3}$. Tables 3 and 4 also show that the APDCA requires fewer iteration steps than the other two algorithms. Specifically, from Table 3, the iteration step of APDCA is about 72% of GIST for the parameter $\lambda = 5 \times e^{-4}$. From Table 4, the iteration step of APDCA is about 83% of GIST and is about 8.6% of PDCA for the parameter $\lambda = 1 \times e^{-3}$. Meanwhile, Tables 3 and 4 also show that the solution given by APDCA is more sparse than that given by GIST and PDCA.

5. Conclusions

In this paper, we propose an accelerated proximal point algorithm for the difference of convex optimization problem by combining the extrapolation technique with the proximal difference of the convex algorithm. By making full use of the special structure of difference of convex decomposition and the information of stepsize, we prove that the proposed algorithm converges at rate of $O(1/k^2)$ under milder conditions. The given numerical experiments show the superiority of the proposed algorithm to some existing algorithms.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors' Contributions

The authors equally contributed to this paper and read and approved the final manuscript.

Acknowledgments

This project was supported by the Natural Science Foundation of China (grants nos. 11801309, 11901343, and 12071249).

References

[1] A. Alvarado, G. Scutari, and J.-S. Pang, "A new decomposition method for multiuser DC-programming and its applications," *IEEE Transactions on Signal Processing*, vol. 62, no. 11, pp. 2984–2998, 2014.

[2] M. Sanjabi, M. Razaviyayn, and Z.-Q. Luo, "Optimal joint base station assignment and beamforming for heterogeneous networks," *IEEE Transactions on Signal Processing*, vol. 62, no. 8, pp. 1950–1961, 2014.

[3] P. Yin, Y. Lou, Q. He, and J. Xin, "Minimization of ℓ_{1-2} for compressed sensing," *SIAM Journal on Scientific Computing*, vol. 37, no. 1, pp. A536–A563, 2015.

[4] G. Wang, Y. Wang, and Y. Wang, "Some Ostrowski-type bound estimations of spectral radius for weakly irreducible nonnegative tensors," *Linear and Multilinear Algebra*, vol. 68, no. 9, pp. 1817–1834, 2020.

[5] G. Wang, G. Zhou, G. Zhou, and L. Caccetta, "Z-eigenvalue inclusion theorems for tensors," *Discrete & Continuous Dynamical Systems-B*, vol. 22, no. 1, pp. 187–198, 2017.

[6] G. Wang, Y. Zhang, and Y. Zhang, "\begin {document} $ Z $\end {document} -eigenvalue exclusion theorems for tensors," *Journal of Industrial & Management Optimization*, vol. 16, no. 4, pp. 1987–1998, 2020.

[7] W. De Oliveira, "Proximal bundle methods for nonsmooth DC programming," *Journal of Global Optimization*, vol. 75, no. 2, pp. 523–563, 2019.

[8] D. Feng, M. Sun, and X. Wang, "A family of conjugate gradient methods for large-scale nonlinear equations," *Journal of Inequalities and Applications*, vol. 236, 2017.

[9] H. A. Le Thi and T. Pham Dinh, "DC programming in communication systems: challenging problems and methods," *Vietnam Journal of Computer Science*, vol. 1, no. 1, pp. 15–28, 2014.

[10] H. A. Le Thi and T. Pham Dinh, "DC programming and DCA: thirty years of developments," *Mathematical Programming*, vol. 169, no. 3, pp. 5–68, 2018.

[11] Z. Lu and Z. Zhou, "Nonmonotone enhanced proximal DC algorithms for a class of structured nonsmooth DC programming," *SIAM Journal on Optimization*, vol. 29, no. 4, pp. 2725–2752, 2019.

[12] Y. Lou, T. Zeng, S. Osher, and J. Xin, "A weighted difference of anisotropic and isotropic total variation model for image processing," *SIAM Journal on Imaging Sciences*, vol. 8, no. 3, pp. 1798–1823, 2015.

[13] X. Wang, "Alternating proximal penalization algorithm for the modified multiple-sets split feasibility problems," *Journal of Inequalities and Applications*, vol. 2018, p. 48, 2018.

[14] D. T. Pham and H. A. Le Thi, "Convex analysis approach to DC programming:theory, algorithms and applications," *Acta Mathematica Vietnamica*, vol. 22, pp. 289–355, 1997.

[15] D. T. Pham and H. A. Le Thi, "A D. C. Optimization algorithm for solving the trust-region subproblem," *SIAM Journal on Optimization*, vol. 8, pp. 476–505, 1998.

[16] J.-Y. Gotoh, A. Takeda, and K. Tono, "DC formulations and algorithms for sparse optimization problems," *Mathematical Programming*, vol. 169, no. 1, pp. 141–176, 2018.

[17] B. O' Donoghue and E. J. Candes, "Adaptive restart for accelerated gradient schemes," *Foundations of Computational Mathematics*, vol. 15, pp. 715–732, 2015.

[18] H. A. Le Thi, V. N. Huynh, and T. Pham Dinh, "Convergence analysis of difference-of-convex algorithm with subanalytic data," *Journal of Optimization Theory and Applications*, vol. 179, no. 1, pp. 103–126, 2018.

[19] X. Wang, Y. Zhang, H. Chen, and X. Kou, "Convergence rate analysis of the proximal difference of convex algorithm," *Mathematical Problem in Engineering*, vol. 2021, Article ID 5629868, 5 pages, 2021.

[20] Y. Nesterov, "A method of solving a convex programming problem with convergence rate $O(1/k^2)$," *Proceedings of the USSR Academy of Sciences*, vol. 269, pp. 543–547, 1983.

[21] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, Springer, Berlin, Germany, 2004.

[22] Y. Nesterov, "Dual extrapolation and its applications to solving variational inequalities and related problems," *Mathematical Programming*, vol. 109, no. 2-3, pp. 319–344, 2007.

[23] Y. Nesterov, "Gradient methods for minimizing composite functions," *Mathematical Programming*, vol. 140, no. 1, pp. 125–161, 2013.

[24] X. Wang, Y. Wang, Y. Wang, and G. Wang, "An accelerated augmented Lagrangian method for multi-criteria optimization problem," *Journal of Industrial & Management Optimization*, vol. 16, no. 1, pp. 1–9, 2020.

[25] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[26] A. Moudafi and A. Gibali, "$l_1 - l_2$ regularization of split feasibility problems," *Numerical Algorithms*, vol. 78, no. 3, pp. 739–757, 2018.

[27] B. Wen, X. Chen, and T. K. Pong, "A proximal difference-of-convex algorithm with extrapolation," *Computational Optimization and Applications*, vol. 69, no. 2, pp. 297–324, 2018.

[28] F. Facchinei and J. Pang, *Finite Dimensional Variational Inequalities and Complementarity Problems*, Springer, Berlin, Germany, 2003.

[29] R. T. Rockafellar and R. J.-B. Wets, *Variational Analysis*, Springer, Berlin, Germany, 1998.

[30] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, Cambridge, UK, 2004.

[31] D. Goldfarb, S. Ma, and K. Scheinberg, "Fast alternating linearization methods for minimizing the sum of two convex functions," *Mathematical Programming*, vol. 141, no. 1-2, pp. 349–382, 2013.

[32] P. Gong, C. Zhang, Z. Lu, J. Huang, and J. Ye, "A general iteraitve shrinkage and thresholding algorithm for nonconvex regularized optimization problems," in *Proceedings of the 2013 30 th International Conference on Machine Learning*, Atlanta, GA, USA, 2013.